IIT/K COMPUTER NETWORK—TDC-316 HARDWARE INTERFACE

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

By

N. S. NARAYANAN

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JULY, 1979

TO

SSB

EE-1979-M-NAR-COM

CERTIFICATE

This is to certify that the work on 'IIT/K Computer Network — TDC - 316 Hardware Interface' has been carried out under our supervision and guidance and this has not been submitted elsewhere for a degree.

(A.S. Sethi)
- Lecturer
Computer Science,
Indian Institute of Technology
Kanpur-208016.

(V. Rajaraman)
- Professor
Department of Electrical Engg.
and
Computer Science,
Indian Institute of Technology,
Kanpur-208016.

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to Dr. V. Rajaraman for his invaluable suggestions and guidance. My gratitude to Dr. A.S. Sethi cannot be expressed in words - he was with us throughout, encouraging, correcting mistakes, giving suggestions - as only he can.

I wish to thank all the engineers in ACES for their suggestions and help-in particular to Mr. Arjun Raman, Mr. K.S. Ananthakrishnan and Mr. S. Manoharan. I thank my colleagues - Debasis Das, Krishna, Pawagi, Jain and Sanjeev - to them I owe the success of this project. The dreary life of hardware development was made a pleasant and unforgettable experience by Aravanan, Prabhakar and Parameswar. My thanks to them.

I thank C.M. Abraham for the superb typing job he has done.

July, 1979. - Narayanan

CONTENTS

FIGURES

CHAPTER 5

ABSTRACT

A system is proposed to link up DEC-1090, TDC-316 and IBM-1800 at the IIT/K Computer Centre. These three machines are connected in a star configuration which uses a micro-computer system (ECIL's MICRO-78) as the central switch. The connections are full-duplex with a data rate of 10 Kbauds. The network is designed for resource sharing and for experimental work in various aspects of computer networks. Flexibility is the prime concern in the design of the system. This report describes the overall system design, line protocol design and the hardware implementation of the line protocols on the TDC side.

CHAPTER 1

INTRODUCTION

A computer network is a complex collection of many types of resources which include data bases, programs, operating systems and special purpose hardware all of which are capable of being accessed from any other resource in the network [KAHN-72]. The general trend towards the formation of computer networks started in the early seventies with the U.S. Department of Defence's ARPANET. ARPA [KLEI] aimed at reliable sharing of specialised computer resources. After the ARPANET, many other networks became functional like the SITA [BRAN-72] for airlines reservations, National Physical Laboratory's [DAVI-68] network, University of Hawaii's ALOHA [ABRA and KUO] and the French CYCLADES [POUZ-74].

The applications of computer networks are many, the most important being sharing of various resources. Another application is load-sharing, where, a computer which is idle could take up some load of an already saturated machine. A third application is distribution of and access to information from remote locations - a typical practical situation is in the airlines reservations systems. Over and above all this, there are special purpose networks which are used in banks, in retail stores as point-of-sale terminals, police networks etc.

1.1 Basic Concepts :

A computer network may be viewed as a set of nodes connected together by edges (channels). Information is transferred in a network from one node to another in units called messages. The path followed by a message from the source node to the destination node is determined as it passes from node to node through the network; no continuous link is established from the source to the destination. This form of message transfer is called message switching and contrasts with circuit switching used in telephone networks, where a physical path is set up in advance of the message transfer. Some networks break up a large message into smaller units called packets. The individual packets are then sent through the network and are reassembled at the destination to get the original message. This is called packet switching [CROW-75]. To carry out the various tasks associated with information transfer in a message switched or packet-switched network, the network usually employs additional computing power in the form of communications processors called interface processors [ORNS-72]. These processors together with the communication lines are called the 'communications subnetwork'. The users are not bothered about the existence of this subnetwork. The interface processors form the interface between the subnetwork and the main computers called 'Hosts' [WALD-75].

The nodes of the computer network may be interconnected in various ways. Some of the important configurations are star, loop, fully connected, tree,distributed and hierarchical. The configuration chosen for a network has a significant effect on the design of the network and its performance. For local networks covering a small geographical region, the star, loop and distributed configurations are commonly employed [KAHN-72]. More detailed discussion on the characteristics of these configurations is given in Chapter 2.

1.2 Protocols and their Functions :

For proper communication between two processors, some communication procedures called protocols [FRAS-76] are necessary which govern the exchange of data. Thus a protocol is the set of agreements between two processors on the format, and relative timings of the messages to be exchanged. When users exchange information, the messages go through many levels of protocols to reach their destination. This is because the protocols are usually arranged in a hierarchy to facilitate implementation and understanding. Each level of protocol in the hierarchy has a definite function to perform. The lower levels are concerned with details of communication while the higher levels are function oriented [FRAS-76]. The lowest levels of protocols are called 'line protocols' and describe mainly the communication between two processors connected by a physical line.

The main functions of the line protocols are to maintain synchronisation between the sender and receiver, initiating message transfer, terminating message transfer, acknowledgement mechanisms to indicate correct or erroneous reception of messages, error detection and recovery techniques [GRAY-72 and STUT-72].

1.3 IIT/K Computer Network :

Although computer networks is a fast developing field, not much work has been done in this area in our country. TIFR, Bombay have developed an experimental network which links up TDC-316s to DEC-1090. AIR-INDIA is planning a big network for airlines-reservations with a UNIVAC as the central node and various terminals all over the country, and abroad. ISRO is planning a data network which links up their computers at Ahmedabad, Trivandrum, Bangalore and Sriharikotah.

In this background, the Computer Centre at IIT Kanpur has taken up a project involving the construction of a small, local computer network. This network is both experimental and utilitarian. It will provide the practical experience needed in the design and implementation of a network and also serve as a vehicle for experimental studies in network flow control, routing, protocols and performance evaluation. However, the network also provides services to the users of the Computer Centre and is planned to ultimately become an essential facility of the Centre.

The Computer Centre currently has three machines available to the general user : the DEC-1090 (which recently replaced the IBM-7044/1401 system), the TDC-316 and IBM-1800. Although the DEC-1090 provides the central computation facility, its utility can be greatly enhanced by linking all the three machines in a Computer Network. DEC users would be able to access the card punch of IBM-1800, a device not available on the DEC system. Moreover, the network can be used to convert old 7-track magnetic tapes to the 9-track tapes used in the DEC, by using the 7-track tape drive unit of IBM-1800. The TDC-316 is currently planned as a remote job entry station for the DEC system. There is also a plan to use it as a concentrator for terminals, thus allowing easy system expansion.

To be able to satisfy these diverse requirements, it is necessary that the network should be flexible and not limited in any way by the design of the initial hardware. Because of resource limitations, the hardware should also be cheap and simple. Thus simplicity, economy and flexibility became the most important design goals of the network. For reasons to be discussed in the next chapter, a star configuration was chosen for the network and a microcomputer, MICRO-78 produced by ECIL, Hyderabad was chosen as the central node of the network. This thesis describes the TDC-316 hardware

interface design and implementation. For MICRO-78 and IBM-1800 interfaces, the reader is requested to consult the companion theses [DDAS-79 and CVK-79].

Chapter 2 of this thesis, explains the system configuration the design goals, the various constraints, and decisions taken considering all these factors. In Chapter 3, we try to retrace the path we adopted to arrive at simple and flexible line control procedures. Chapters 4 and 5 describe in detail the hardware design and implementation of the TDC-316 sending and receiving interfaces respectively. Chapter 6 gives a think-over on the complete system and some discussions on what could be done with the present work as a foundation.

CHAPTER 2

SYSTEM CONFIGURATION

When a large system such as a computer network is designed, it is always good to divide the system into various logical levels. This layered design approach enables one to concentrate on specific areas so that fewer mistakes are made. Further it defines the extent of each layer and simplifies the design. This was the approach taken in the present project. The steps in the system design are as follows :

1) Define the aims and purposes of the network
2) Configure the system
3) Define the line control procedures-the basic level of control
4) Check the expected hardware complexity
5) Repeat steps 1,2 and 3 until the aims listed in step 1 are satisfied
6) Design the requisite software which interfaces the system to the monitor
7) Design the overall network control procedures and monitor routines - essentially a Network Operating System.

2.1 Philosophy of System Design :

The design aims were the following :

1) Since the system is going to be used in a university environment, flexibility rather than efficiency should be the main aim. This in turn means that, the maximum control should be given to software. The hardware design should impose the minimum possible limitations on the options open to the software designer.

2) The hardware should be simple, general and logically similar in all the machines. This would reduce the cost also.

3) The feasibility of using a microcomputer for switching is to be studied and verified.

4) The system should be useful, and it should be possible for the users of any machine to get access to system facilities and peripherals of the other two machines.

5) The overall structure of the hardware and software should be simple to understand.

2.2 Network Configuration :

Although computer networks have a variety of configurations, three possible configurations are most attractive for a small network like ours.

(a) Star (Fig. 2.1) - This configuration has a central node to which all other nodes are directly connected. The advantages are: (1) it makes the overall control easier - the control is possible at the central node.

(ii) It makes the system conceptually simpler to understand. A disadvantage is that, the star network may have large line lengths if the nodes are far from the central node, but in our case, since distances involved are small, the line lengths will not be excessive. Another disadvantage is that the system is too dependent on the central node and hence the reliability is quite low. A system using multiple microprocessors at the switching centre would be an interesting thought - this might push up the data rates and increase the reliability.

(b) Data Loops (Fig. 2.2) - Data loops are operational in a few Universities. The advantage of data-loops is, complex routing decisions need not be taken since the path is only one. The main problem with data loops is the relatively complex line control procedures which need sophisticated hardware. Designing a flexible system using data loops is fairly complicated and was not thought advisable with our limited resources.

(c) Fully or partially connected network (Fig. 2.3) is not called for since reliability is not our major consideration. The data rates needed for our proposed applications are also not large enough to justify the higher cost and complexity of such a network.

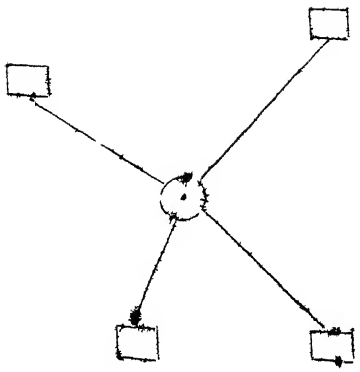


Fig. 2.1 Star Configuration*

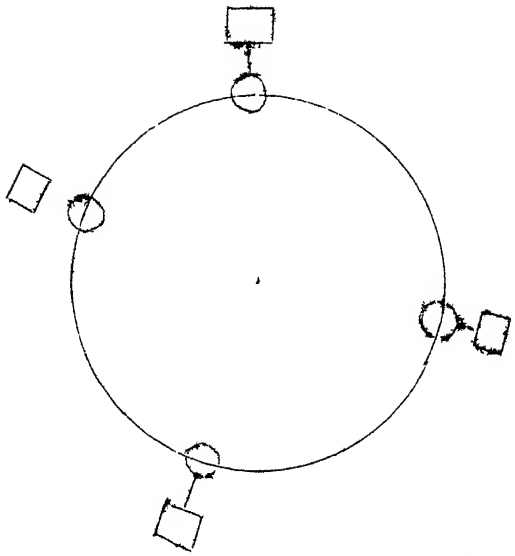
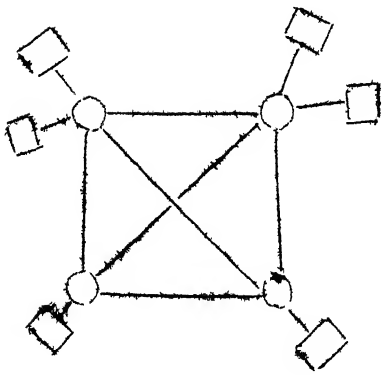
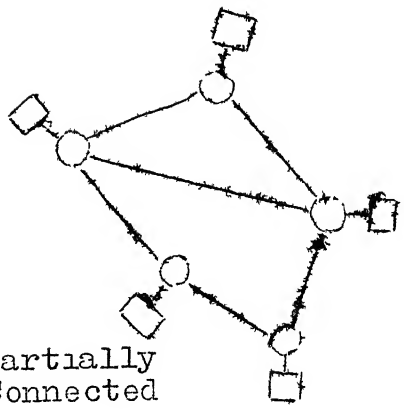


Fig. 2.2 Data-loops*



a) Fully Connected



b) Partially Connected

Fig. 2.3 Connected Networks*

* □ - Hosts , ○ - Nodes

For all these reasons, a star network was chosen with the IBM-1800, TDC-316 and DEC-1090 as the three outer nodes and MICRO-78 as the switching centre (Fig. 2.4).

Present day microprocessors are providing more functions and are giving facilities which earlier minicomputers were capable of. This was the reason why we decided to try out a microprocessor as the interface processor. Since MICRO-78 was about the only commercial system with a floppy disk facility, easily available in our country, it was decided to try it out.

2.3 Data Transfer Modes :

Having chosen the star configuration, the choice of full duplex serial data transmission between any pair of nodes is almost axiomatic since it is the simplest, cheapest and most efficient form of communication for the distances involved. Once full duplex lines are agreed upon, we can now concentrate on the subsystem shown in (Fig. 2.5). We have two processors connected together by two cables which transfer data in opposite directions. As far as any Pc is concerned it sees a hardware sending interface and a receiving interface.

Data transfer between the Pc and the interfaces are possible in two modes (i) by Interrupt, (ii) by DMA. The choice between the two depends on the ease of implementation in hardware and efficiency of transfer. In the interrupt mode,

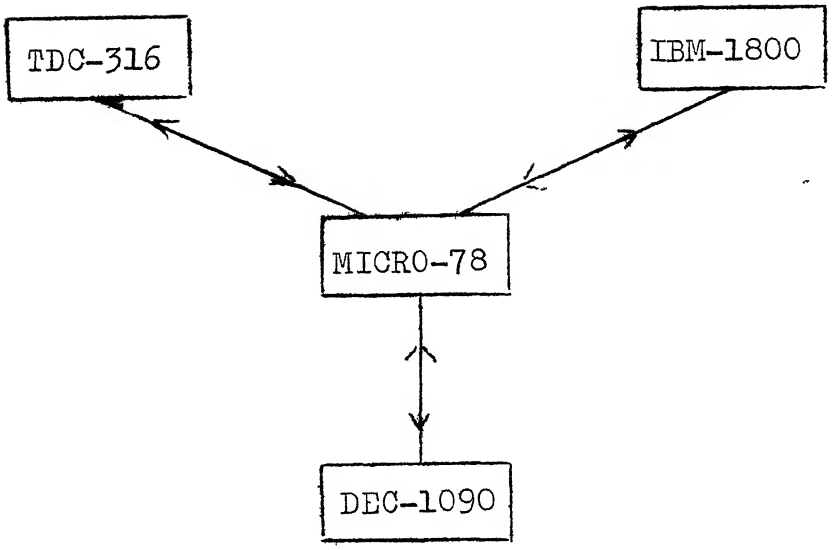
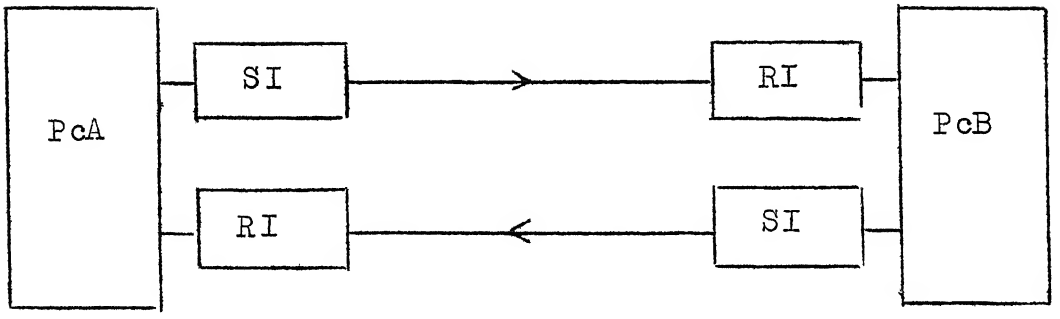


Fig. 2.4 System Configuration



SI - Sending Interface RI - Receiving Interface

Fig. 2.5 Dual Processor Subsystem

the Pc is interrupted by the interface, for each byte sent/ received. If the interface has a single byte buffer, the time available to service the interrupt is equal to the transmission time of one byte. In DMA mode, the byte is obtained from/ deposited into the memory without processor intervention. Since the overhead for DMA is much less, it can support very high data rates typically 250 Kbits/sec. But it is to be remembered that, what the Pc does through software in the interrupt routine has to be done by hardware in the DMA mode. DMA interface is much more complicated than an interrupt interface in terms of hardware involved, in the case of TDC-316 and MICRO-78 (In the case of IBM-1800, special digital input/output groups are available and data could be transmitted by a special DMA type of operation called Data Channel). So to save on hardware, it was decided to have interrupt interfaces for TDC-316 and MICRO-78, and data channel transfer for IBM-1800.

Since DEC-10 protocols specify a data rate of 9.6 Kbauds, it was decided to accept that as standard. All the interfaces were designed to transmit at 10 Kbauds for it was easy to generate a stable 10 KHz clock. The only thing which has to be checked is whether MICRO-78 can handle this data rate. MICRO-78 has to handle six interfaces - three sending interfaces and three receiving interfaces. So it might have to

handle a maximum of six interrupts, in the transmission time of a byte, if data overrun is not to occur. Assuming 8 bit byte plus 1 bit for parity, the time to receive/send one byte is 900 μsecs. In 900 μsecs, six interrupts have to be serviced. So each interrupt service routine must take less than 150 μsecs. At 2 μsec average instruction cycle time, the MICRO-78 could execute about 75 instructions in this time which, was considered more than adequate for the present applications.

2.4 An Overview of the Complete System :

From the above discussions, the following characteristics emerge for our network.

* A star network with TDC-316, IBM-1800 and DEC-1090 at the three nodes and a MICRO-78 as the central switch is envisaged.
* Full duplex transmission at 10 Kbauds chosen.
* TDC-316 and MICRO-78 are going to operate in byte interrupt mode and IBM-1800 is going to use the data channel.

The next step is to devise a set of control procedures for transmitting and receiving data. Since this depends on the machine characteristics, it is now necessary to look more closely at the machine-TDC-316.

2.5 TDC-316 Characteristics :

2.5.0 Introduction :

The TDC-316 configuration at IIT, Kanpur, has one disk, line printer and card-reader as peripherals. The system has two buses called the M-Bus and the I-Bus. Otherwise, the organisation is very similar to the Unibus structure of PDP-11. In fact, the I-bus is functionally the same as the unibus.

Only memory is connected to the M-Bus while both memory and I/O devices including disks may be connected to the I-Bus. The bus is at any time under the control of a master who communicates with a slave. The master-slave relationship is dynamic and different devices may become master and slave at different times. However, memory can never become the bus-master.

The system has 56 Kbytes (28K words) of core memory. The memory is word oriented (16 bit word) but is byte addressable. Although the amount of physical memory is only 28K words, the addressable memory space is 32K words. The last 4K words of this space are used for I/O registers and processor registers. The machine uses memory mapped I/O. There are no separate I/O instructions; any instruction which can operate on the memory can also operate on the I/O registers, provided proper

hardware is connected to the addressed I/O registers. The function of the I/O registers is device dependent and any type of hardware register can be used: read only, write only or read/write; it could also be a register whose individual bits have any or all these alternatives.

2.5.1 Transfer of the Bus [TDCSYSI] : A device requests to be a bus master for

(1) direct data transfer to and from memory (Non-processor request), or

(2) interrupting the program and forcing the processor to branch to an interrupt routine.

When the Pc is executing a routine, it executes it at a certain priority level, ranging from 0 to 7. (Level 0 has the lowest priority). The processor priority is defined by bits 5,6 and 7 of the Processor Status register (PS) and can be modified by software.

The interrupting device can request for the bus at any level 1 - 7. Correspondingly, there are 7 bus request lines BR1 - BR7, and 7 Bus Grant lines, BG1 - BG7. A number of devices may be daisy-chained on the same level.

2.5.2 Interrupt Transactions for TDC-316 [TDCSYSI] :

When a device generates an interrupt, the following events take place.

(1) The device first generates a bus request and depending on the present processor priority, and the level at which the bus request is made, the Pc grants the bus to the device.

(2) The device then puts the interrupt vector address on the I-Bus data lines, and asserts INTR and BBSY signals and drops SACK. The interrupt vector consists of two words in the memory - the first contains the address of the interrupt routine and the second contains the processor status when the interrupt routine is executed. The interrupt vector address is a double word address, with the high byte all zeros and the least significant two bits also as zeros. Thus the interrupt vector may be anywhere between 0-374/8, and its address has to be a multiple of 4.

(3) The Pc on getting the interrupt vector, stacks its present PS and PC on the supervisor stack and fetches the new PC and PS from the double-word pointed to by the interrupt vector.

It is possible to have multiple levels of interrupt. When an interrupt of priority higher than that of the present one occurs, the Pc proceeds as before stacking its present PS and PC on the stack and taking the new PC and PS from the locations pointed to by the interrupt vector.

CHAPTER 3

LINE CONTROL PROCEDURES

Line Control Procedures or Data Link Controls (DLCs) are hardware and software protocols whose functions are (i) to establish a connection between the sender and the receiver (ii) to synchronise the two parties (iii) to pass messages and (iv) to recognise and correct the errors during transmission [GRAY-72]. Essentially, the DLCs provide a variety of data communication facilities to send and receive messages.

The main characteristic of the DLC is that, it acquires and maintains synchronisation [DAVI and BARB] in order to pass message between higher level users. Since a message may consist of hundreds of bits, serialisation of the message is essential. The major problem is to recover the clock at the receiving side from a random bit sequence. Once this is done, it should be possible to recover the characters also. There are two techniques available for this (i) In asynchronous communication systems, a start-bit signifies the start of the byte and a stop-bit signifies its end. (ii) In synchronous systems, a special character, called SYN, is transmitted and the instant SYN is decoded, the receiver gets synchronised with the transmitter. Once byte synchronisation is done, subsequent data bytes could be detected properly since the clocks are synchronised and data comes continuously.

A channel is never error-free, and various errors might creep in corrupting the message. These could be bit or burst errors. To detect errors, redundancy has to be built into the transmitted information. The coming sections describe a set of protocols and illustrate how bit, byte and message synchronisation and error-checking and recovery take place in the present system.

3.1 Network Line Protocols :

Here we consider how two processors connected together by full-duplex cables, each having a sending interface which transmits data and a receiving interface that receives data. (Fig. 2.5).

In general 8-bit ASCII characters are sent over the line. Characters transmitted are of two types - those which are data and those which are control information. Since there is no separate control line, the only way to distinguish between data and control characters is by observing what is received at what time.

It is very important that we should not impose any restrictions on the bit patterns of the data. Even if the data character to be transmitted matches a control character, there should not be any ambiguity. So depending on the state of the interface, it should be able to decide whether the character received is data or control and take appropriate

action. In computer jargon, this is termed data transparency.

Another consideration is the length (number of bytes) of the packet. For flexibility we leave this to the software. So hardware does not fix the length of the packet.

The three control characters which are recognised by the interface are SOM(01/16), SYN(26/16) and ACK(7C/16). When the line is idle, the sending interface keeps sending SYN over the line. Once the bit pattern of SYN is recognised at the receiving interface, a counter is initialised, which counts the subsequent bits. The receiving interface also sets a flag denoting byte synchronisation is achieved. The counter is driven by a clock which is derived from the data and thus it is synchronous with the input bit stream. This is the way the bit and byte synchronisation is done. Once the bit and byte synchronisation is achieved, the interfaces are at their idle states and are in a position to receive the message when it comes through.

The only other mode in which the line could be is the data transfer mode. This is signified by the reception of SOM or ACK character at the receiving interface. While in the idle mode, if the receiving interface receives any character other than SOM or ACK, the synchronisation flag is reset and

the line is idle till the next SYN character arrives. No data reception is allowed if the sync flag is not set at the receiving interface.

When in the synchronised state, a SOM or ACK is received, a flag is set at the receiving interface, showing that a message is starting. Also, it generates an interrupt to the receiving processor. The receiving processor should check the character and see whether it is SOM or ACK and take proper action. As far as the interface is concerned, SOM and ACK are treated alike and signify start of a message.

There is no special character called EOM which could signify the end of message at the receiving interface. The receiving processor gets this and other information from the first few bytes in the transmitted data (header). Once the processor knows how many bytes to expect in the message, message framing is possible by software.

3.2 Error Checking and Recovery :

Attached to each 8-bit byte, is a single parity bit, which keeps the parity of the 9 bits odd. Thus single bit error detection is all that is possible. More advanced parity checking and correction techniques could be implemented in software like checksum error, vertical and horizontal parity

checking etc. If a single bit error is present, it is detected by hardware and sets a flag in the receiving interface.

The system uses a positive acknowledgement scheme. If the processor finds that the data received is proper, it sends a positive acknowledgement over its sending line. If the line is already engaged in sending a packet, the acknowledgement is sent after the present packet. Meanwhile the sending processor waits for a timeout signal from its timer, which it enables at the end of data transmission. If the ACK is received before the timeout period, then the next packet transmission will be initiated. If no ACK is received in the timeout period, then the previous message or packet is assumed lost and the same packet is retransmitted. There is facility to put a 1-bit sequence number in the ACK character and is used in software to identify the packet for which ACK was sent.

The set of line protocols and their hardware implementation are given in the next two chapters.

CHAPTER 4

SENDING INTERFACE

In this chapter we formally define those aspects of the protocols which are to be implemented through hardware in the sending interface. It is most convenient to state the protocols in a hardware design language [CHU], so that direct correspondence between the description and actual hardware can be established.

The TDC-316 processor is connected to two interfaces - one sending interface and one receiving interface. The processor has access to a pair of registers in each interface. One is the Control and Status Register (CSR) which gives the status information of the interface and the other is the buffer (DAT) which is loaded (or read) by the processor when it wants to send (or receive) data. Each of them has an address in the memory space. By setting certain bits of the CSR, the processor controls the status of the interface.

The operation of the sending interface could be briefly summarised as follows :

1. In the idle state the sending interface continuously sends the SYN character over the line.
2. When the Pc wants to send a packet, it loads the first byte of the packet into the DAT buffer and sets a flag (INIT) in the sending CSR.

3. Sending interface loads the byte from DAT into a shift-register (TXRG). This achieves parallel to serial conversion.
4. A 1-bit odd parity is generated and appended to the byte contained in TXRG.
5. The data is transmitted serially from TXRG over the communication line.
6. The loading of TXRG from DAT is followed by the generation of an interrupt to the Pc.
7. The Pc loads the next byte of the packet into DAT and the process continues at step 3.
8. When all the bytes are transmitted, the Pc sets the INIT flag in the CSR off and the interface goes back to the idle state (step 1).

4.1 Hardware Description of the Sending Interface :

Sending CSR is an 8-bit register implemented using D-flip-flops ($\frac{1}{2}$ 7474). The format of CSR is shown in Fig. 4.1.

Sending Protocols :

Definitions :

Register SDAT(0-7), SCSR(0-7), LC, TIMER (0-15)

Shift-register TXRG(0-7)

Clock PP

Character SYN(16/16)

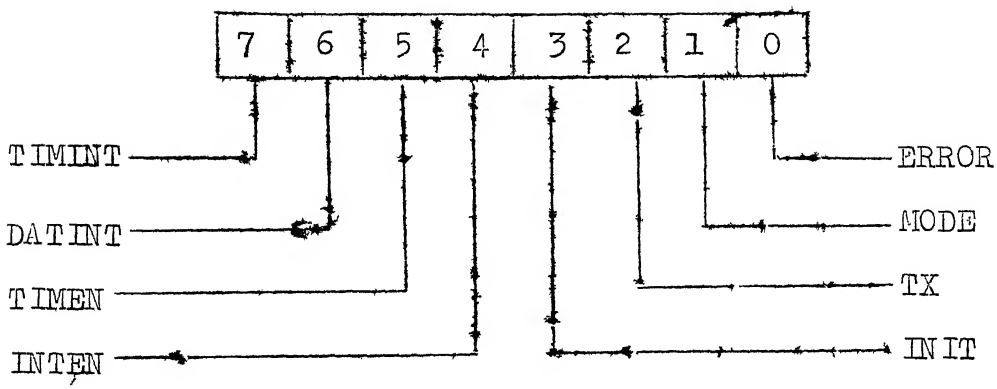


Fig. 4.1 Sending Interface CSR

/($\overline{INIT}$ + $\overline{MODE}$)·(LC=0)·PP/ TXRG ⟵ SYN, MODE ⟵ 1 (S1)

/INIT·MODE·(LC=0)·PP/ TXRG ⟵ SDAT, TX ⟵ 0, DATINT ⟵ 1 (S2)

/(LC≠0)·PP/ Transmit TXRG over the line, shift right TXRG, Decrement LC (S3)

/LC=0·PP/ LC ⟵ 9 (S4)

/INIT·MODE·(LC=0)·$\overline{TX}$·PP/ ERROR ⟵ 1 (S5)

/Data interrupt granted/ DATINT ⟵ 0 (S6)

/SDAT Loaded/TX ⟵ 1 (S7)

/$\overline{TIMEN}$·TIMCK/ Load TIMER (S8)

/TIMEN·TIMER≠0. TIMCK/ Decrement TIMER (S9)

/TIMEN·TIMER=0 · TIMCK/ TIMEN ⟵ 0, TIMINT ⟵ 1 (S10)

/Timer interrupt granted/TIMINT ⟵ 0 (S11)

Note : The information enclosed between // denotes a condition in the interface. Else, all names correspond to registers or bits, Any or some of the conditions // could be satisfied at the interface depending on the bits set.

4.2 Brief Explanation :

When INIT = 0, then SYN is sent over the line. The Pc when it wants to start data transmission loads the data in the buffer (SDAT) and sets the INIT bit. After the present byte transmission, the interface sees the INIT bit set, loads data

into TXRG and generates an interrupt. Along with loading TXRG, it also loads a flip-flop with the parity corresponding to the present 8 bits. The flip-flop is connected to the serial input of TXRG and the 8 bit of the TXRG and the flip-flop form a 9-bit serial shift register which sends data over to the line. Along with loading TXRG, the interface also generates an interrupt so that the Pc may load the next byte in SDAT. The function of CSR bits is as follows :

1. TIMINT : Set when TIMER = 0. Reset by software; when this bit is set it interrupts the Pc if INTEN is set.
2. DATINT : This bit is set by interface after loading TXRG. RESET by loading SDAT.
3. TIMEN : When this bit is set, TIMER is enabled. At the end of the timeout period, TIMINT gets set and TIMEN resets.
4. INTEN : Enables either TIMINT or DATINT to cause interrupt. Set and reset by the Pc depending on whether it wants to enable or disable interrupts.
5. INIT : Set by the Pc when it wants to send data. Reset by Pc at the end of data transmission.
6. MODE : This bit ensures that at least one sync character is sent after a packet is transmitted. At the end of data transfer, Pc sets the MODE to 0. Even if INIT gets set (for the following packet) before transmission

of SYN has started, $\overline{\text{MODE}}$ initiates SYN transmission. After sending one sync character, MODE gets set to 1 and only then does data transmission start.

7. TX : Data handover bit; This is set to 0 by the interface when data has been loaded into the TXRG. When the Pc loads data in SDAT, while executing the interrupt routine, this bit is set. If after the current byte transmission, TX is still 0, it shows that SDAT has not been loaded with new data and that an error in transmission is imminent.

8. ERROR : Shows that an error has occured in transmitted data. i.e. Pc due to some reason or other has not serviced the previous interrupt, till the next one has come. This is determined by looking at TX and checking if it is still zero when data is going to be loaded.

4.3 Sending Interface Hardware Block Diagrams :

The hardware consists of the following functional blocks :

a) Clock Generator
b) Data register and Parallel/Serial Converter
c) Coding Circuit and Line Driver
d) Timer
e) CSR Controls

These blocks are explained in detail in the following sections.

4.4 Clock Generator (Fig. 4.2) :

It consists of a crystal controlled oscillator using LM-375 and a divider chain. The crystal oscillator uses a 2 MHz crystal and division by 50 generates a 40 KHz clock. This is divided by a 2-bit binary counter to get a 10 KHz (A) and a 20 KHz clock (B). The crystal oscillator has high power supply rejection and long term stability. The detailed circuit diagram is given in [Appendix A1].

4.5 Parallel/Serial Converter (Fig. 4.3) :

8-bit data from the Pc is loaded into the 8-bit latch when SDATADDR-SEL and OUTLH lines go high. Depending on whether INIT bit is set in the SCSR, DSEL will be present or not present. DSEL and LOAD signals come at the same time, DSEL being a slightly delayed (of the order of a few gate delays) version of the LOAD. DSEL, if present, selects the data from the multiplexer and loads the TXRG. If DSEL is not present, SYN gets loaded into the TXRG. At the same time PAR flip-flop is also loaded with the odd parity generated output of the parity generator PARGEN (implemented using 74180). All the loading and setting of flip-flops will take place at the negative going edge of LOAD. Since DSEL is delayed with respect to LOAD, it ensures that, at low going edge (active edge) of LOAD, DSEL still holds proper data on the TXRG input. See Fig. 4.4 for the timing diagram.

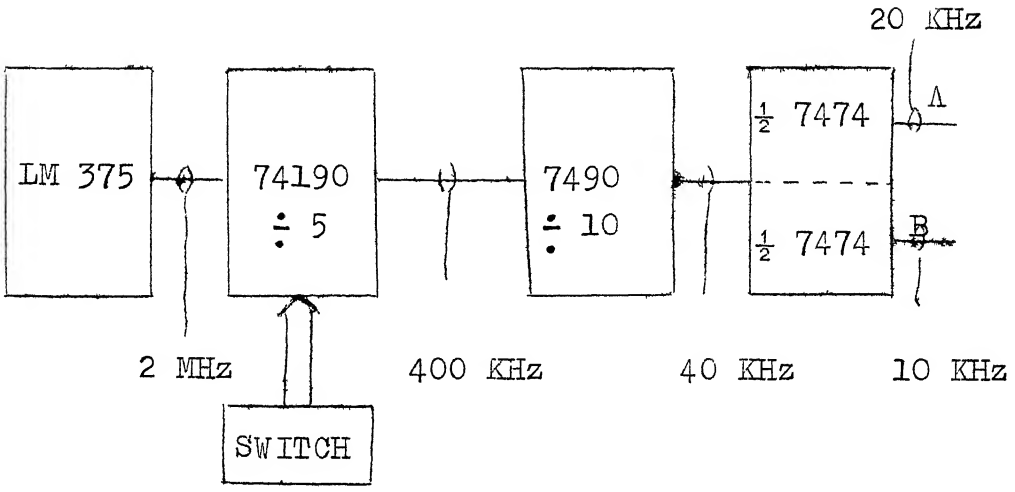


Fig. 4.2 Clock Generator

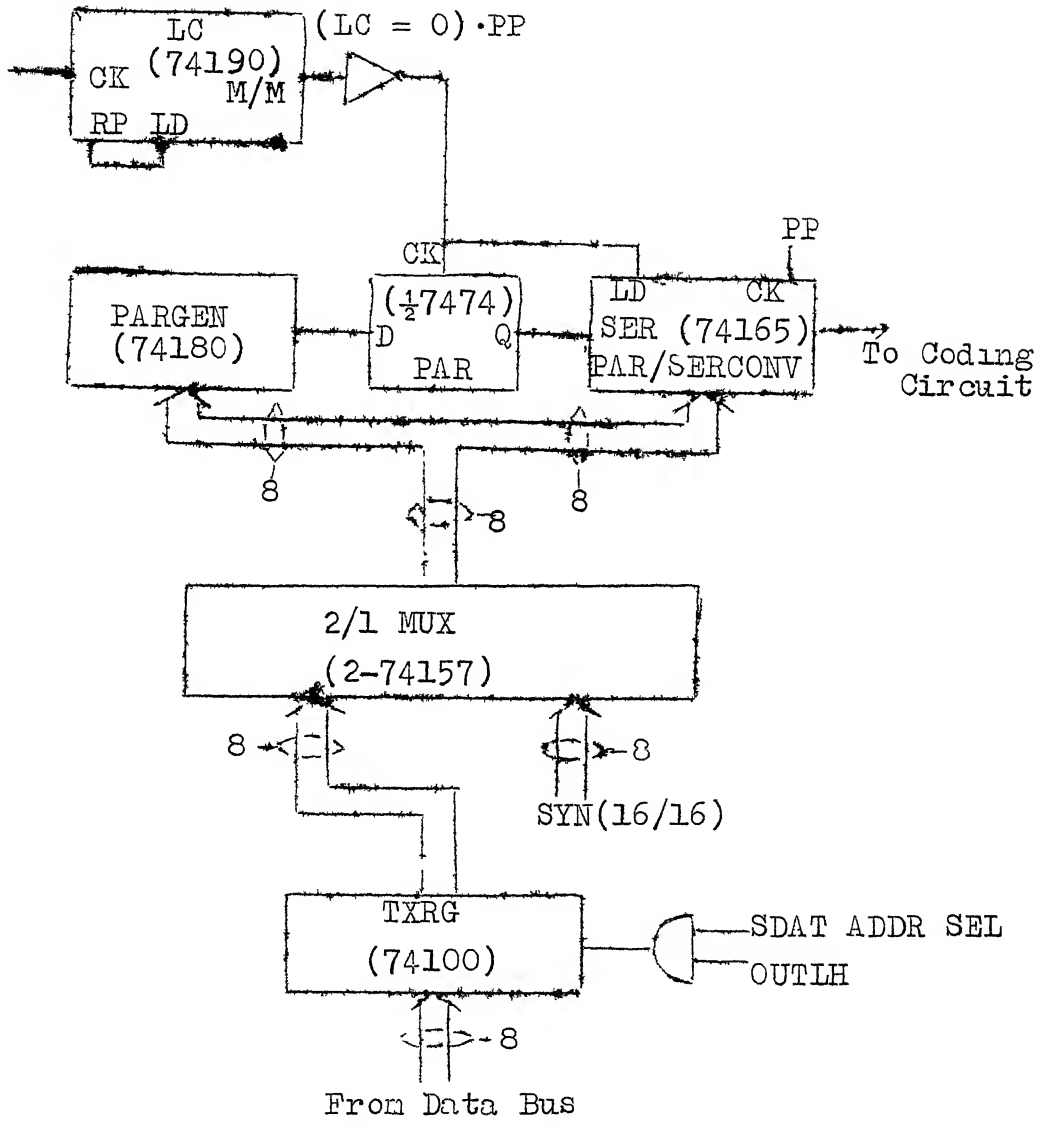


Fig. 4.3 Data Transmission Block Diagram

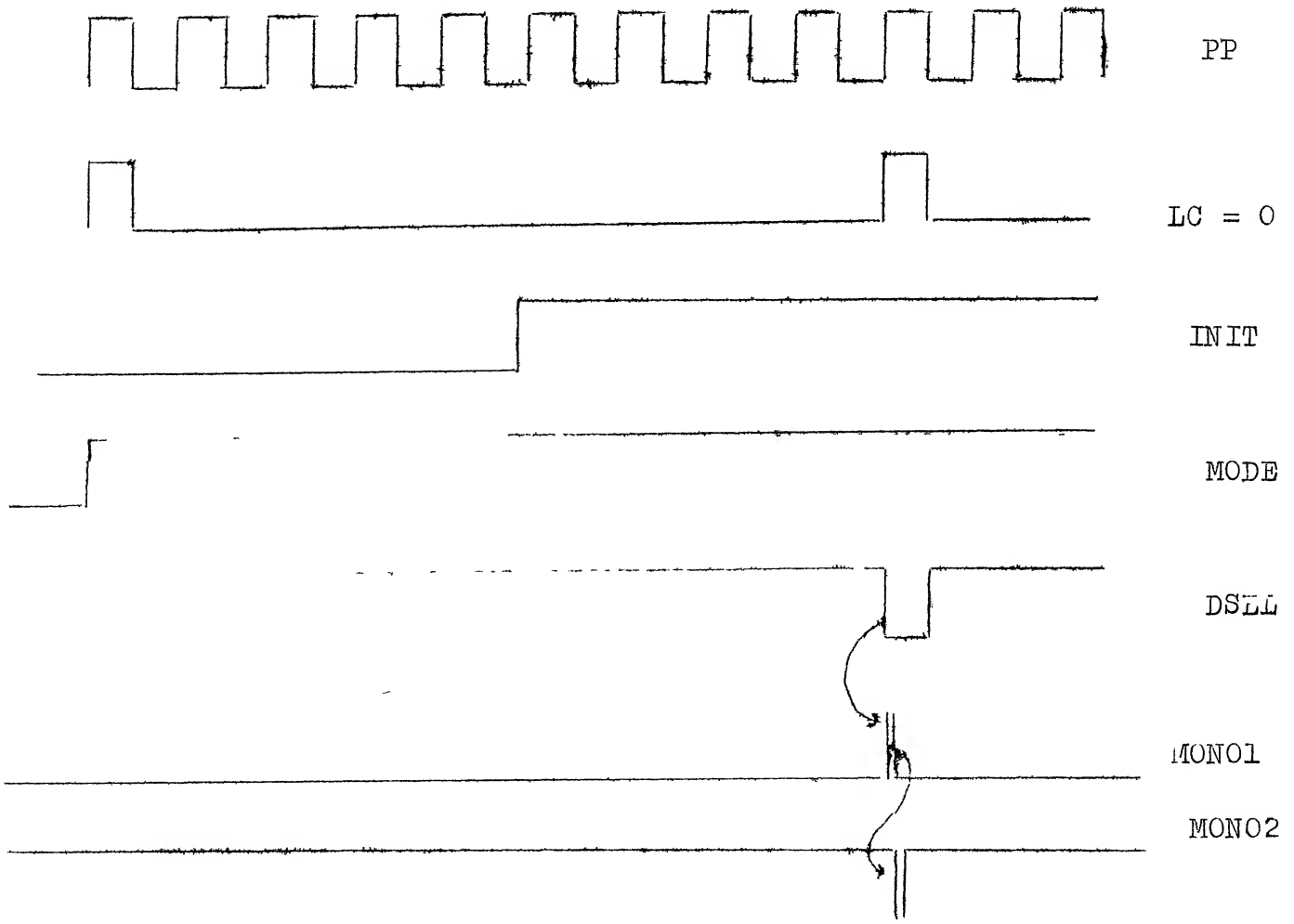


Fig. 4.4 Timing Diagram-Sending Interface

4.6 Coding Circuit and Line Driver :

The serial output of the shift register is coded to get proper transitions at the start of each bit. With these transitions, the clock of the receiving interface is synchronised. The coding used is shown in (Fig. 4.5). It can be seen that the signal for bit 0 remains 1 for the first .25 of the basic clock (B) and is zero for the remaining .75 part. Similarly, the signal for bit 1 is 1 for the first .75 part of B and is 0 for the remaining .25 part. For reasons given in Chapter 5, the data is sent inverted on the line, so that it gives a negative transition at the start of each bit. The logic for coding could be easily derived as

$$\text{Serial output} = I \cdot A + I \cdot B + A \cdot B$$

where A is the 20 KHz signal, B is the 10 KHz signal and I, the serial output of the shift register.

The implementation of the coding circuit is shown in (Fig. 4.6). The final output is inverted and fed to the line driver (Fig. 4.7). The common emitter output of the NPN transistor provides impedance matching and the current drive.

It was observed that, such a line driver could nicely drive a coaxial cable of type 62 A/U of 93 Ohms characteristic impedance. There was no distortion severe enough to be mentioned when driving even about 300 ft of cable.

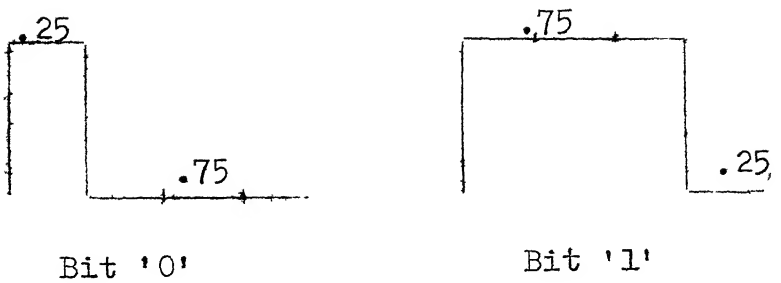


Fig. 4.5 Coded Bits

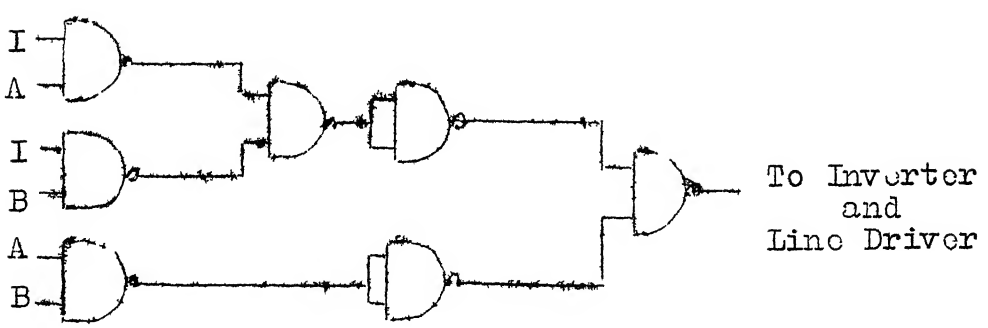


Fig. 4.6 Coding Circuit

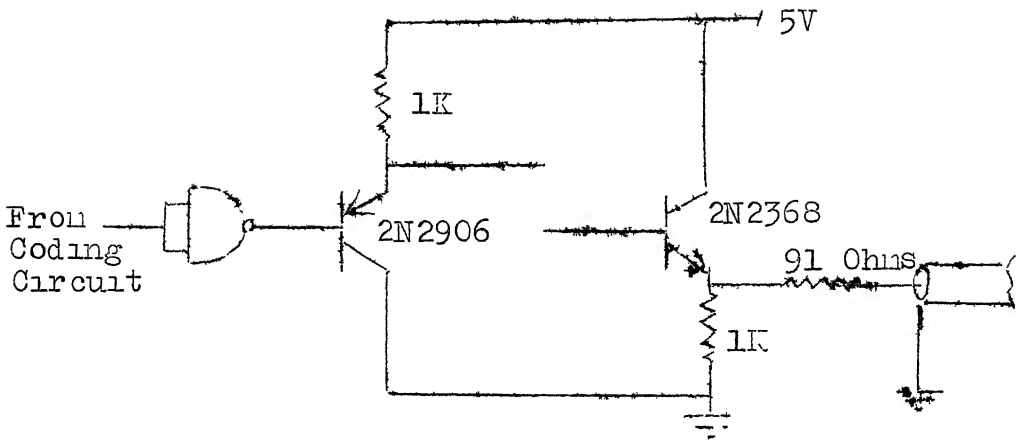


Fig. 4.7 Line Driver

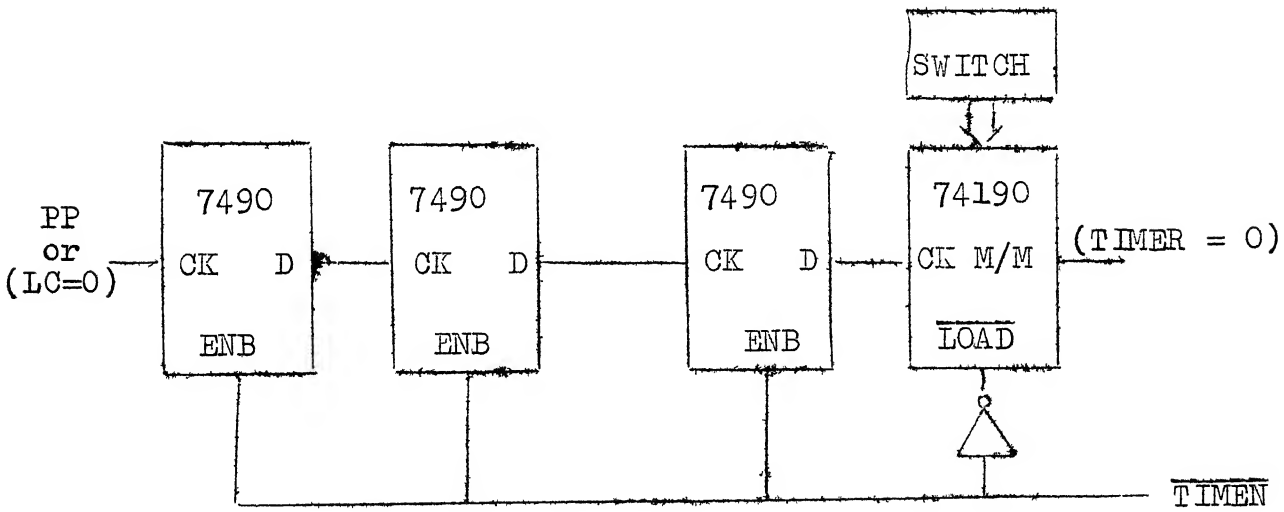


Fig. 4.8 Timer Block Diagram

4.7 Timer :

Since a timing mechanism is needed by the Pc for timing out acknowledgements, and the TDC-316 does not have a real-time clock, it was decided to incorporate a simple timer in the sending interface. The timer uses a series of counters as shown in Fig. 4.8. When the timer is disabled (TIMEN = 0), all the counters are kept loaded. The timer is enabled when the Pc sets the TIMEN bit. The counters start counting and when the count reaches zero, the timer generates an interrupt provided the INTEN bit is set in the SCSR. At the same time, TIMEN bit is reset and the counter returns to the load mode. Actual implementation is by using a 74190 and 7490s. All are connected in cascade, with the D output of one feeding the clock of the next. The first 7490 could be driven by PP or (LC = 0) pulse depending on the back-panel wiring. Hence a maximum time of 1 second or 9 seconds could be obtained. The 74190 counts down while all the rest count up. It is possible to load the 74190 with any value from 0 - 9 via switches and thus any timing from .1 to 1 second at .1 second interval or .9 to 9 seconds at .9 second interval is possible.

4.8 CSR Controls :

Implementation of the 8-bits of the CSR is done by four 7474 dual D-flip-flop chips. Two types of controls are possible (i) Asynchronous controls and (ii) Synchronous controls.

Asynchronous controls preset and clear always have priority over synchronous controls i.e. even when the D and CK terminals are active at any time, if preset or clear tries to do the opposite thing, then the latter will take place. The flip-flops are positive edge triggered. Preset and clear are low enable.

Since Pc should have the ultimate control over the CSR bits, Pc uses asynchronous preset and clear to set/reset CSR bits. The bits should be affected only when the CSR address is selected and the Pc wants to output data, denoted by OUTLH signal. Also, when a bit is to be set, (DATA high), preset should go low and if a bit is to be reset, the clear should go low. The configuration as in (Fig. 4.9) is implemented.

When the Pc wants to read data, it selects the address and puts I NH line high. The logical AND of these two is used to enable an open-collector NAND gate, the output of which is connected to the bus via a buffer, in the transreceiver card.

The above two configurations are kept standard for giving input and taking output from the bus and are available on all the CSR bits. So all CSR bits could be read or written.

Interface controls the CSR bits using the D and CK inputs. (Figs. 4.10A to 4.10 F) show all the individual controls. It is a direct translation of the protocols. The

line counter LC keeps a count of the bits sent over the communication line. It is implemented using a 74190 up/down counter whose RPCK is connected to LOAD. Then MIN/MAX terminal gives (LC = 0) · PP signal. [TTLH].

A. TIMINT Controls (Fig. 4.10A) : At the positive going end of (TIMER = 0) pulse, TIMINT sets. The two inverters between D and CK are for delay.

B. TIMEN Controls (Fig. 4.10B) : INITH is a signal from the machine, when the machine is initialised by pressing the INIT on console or executing CLST instruction. [INIT + (TIMER = 0)] clears TIMEN.

C. INTEN Control (Fig. 4.10C) : All interrupts have to be disabled when the machine is initialised. So INTEN is reset by INITH.

D. DATINT Control (Fig. 4.10D) : DATINT is set at the end of loading TXRG. This is denoted by the DSEL. DATINT gets set by the trailing end of DSEL. DATINT has to be cleared when (i) either Pc wants to clear it, i.e. when $\overline{\text{CSR·ADDRSEL·OUTLH}}$ is 0 or (ii) Pc loads the SDAT buffer i.e. when $\overline{\text{SDAT·ADDRSEL·OUTLH}}$ is 0. Since the clear terminal is low active, we have to AND the two signals.

E. INIT : Interface has no control over the INIT bit.

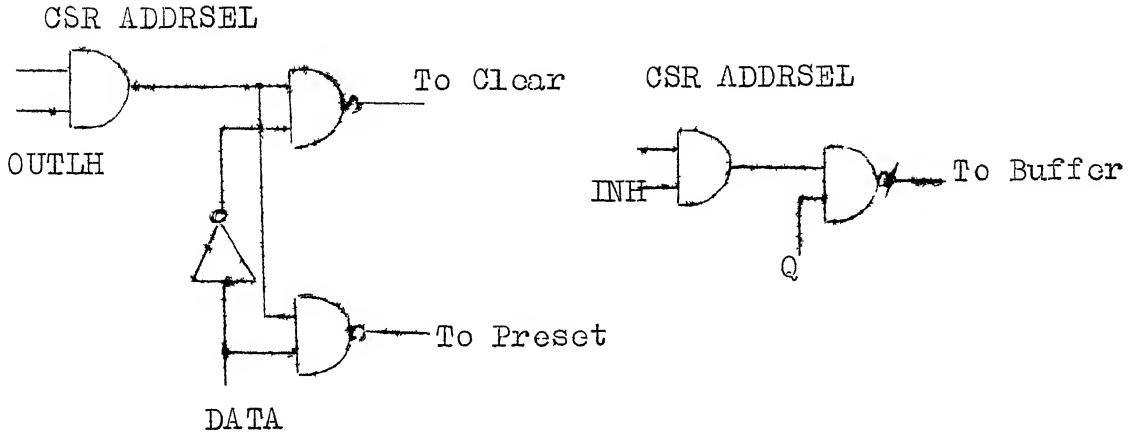
F. MODE Controls (Fig. 4.10E) : MODE is set after sending one sync. character. It is set by the function $(\overline{\text{INIT}} + \overline{\text{MODE}})\cdot$ (LC = 0).

G. ERROR and TX Controls (Fig. 4.10F) : MONO1 and MONO2 enable the DSEL interval to be divided into two. During the first interval, TX is scanned and if $\overline{TX}$ is 1, then ERROR bit gets set. With MONO2 (triggered by MONO1) TX is reset. If $\overline{TX}$ were zero when MONO1 was active then ERROR bit maintains its previous state. In the second interval since MONO1 output is low, even if $\overline{TX}$ goes high, ERROR does not set.

4.9 Interrupt Generation :

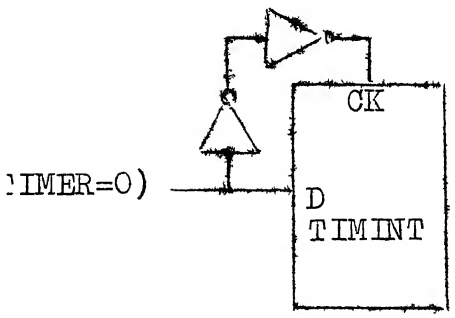
There are two separate sources of interrupt in the sending interface - one is the TIMER (denoted by setting of TIMINT flag) and the other is the data interrupt (denoted by setting of DATINT flag). Pc should be interrupted only when INTEN flag is 1. To save on hardware, both TIMINT and DATINT cause interrupt over the same interrupt line. Which one causes interrupt should be determined by software. So, the logical function for interrupt generation is (DATINT +TIMINT) · INTEN, (Fig. 4.11).

Next chapter describes the receiving interface hardware.



(a) Data Output (b) Data Input

Fig. 4.9 CSR Controls for Pc



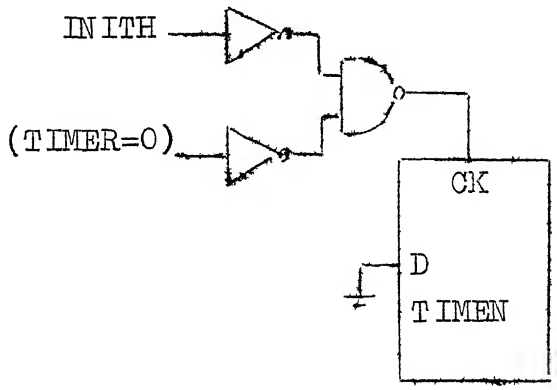
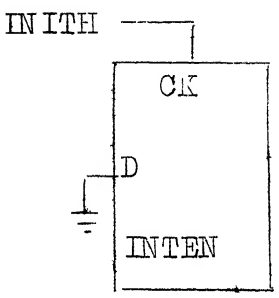


Fig. 4.10A TIMINT Control

Fig.4.10B TIMEN Control



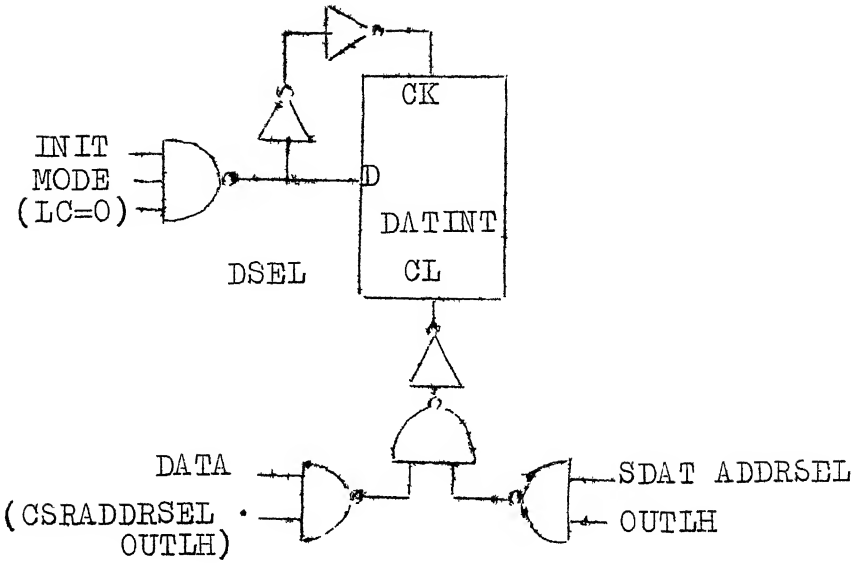


Fig. 4.10C INTEN Control

Fig. 4.10D DATINT Control

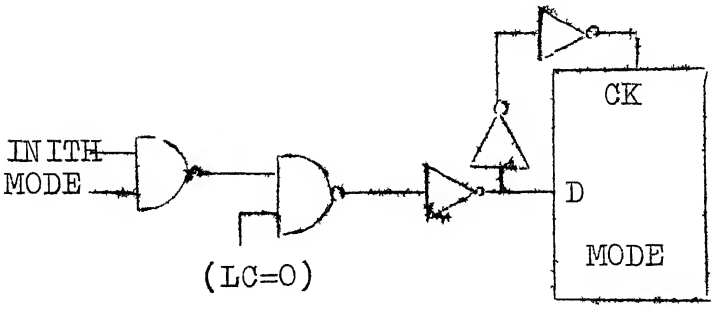


Fig. 4.10E MODE Control

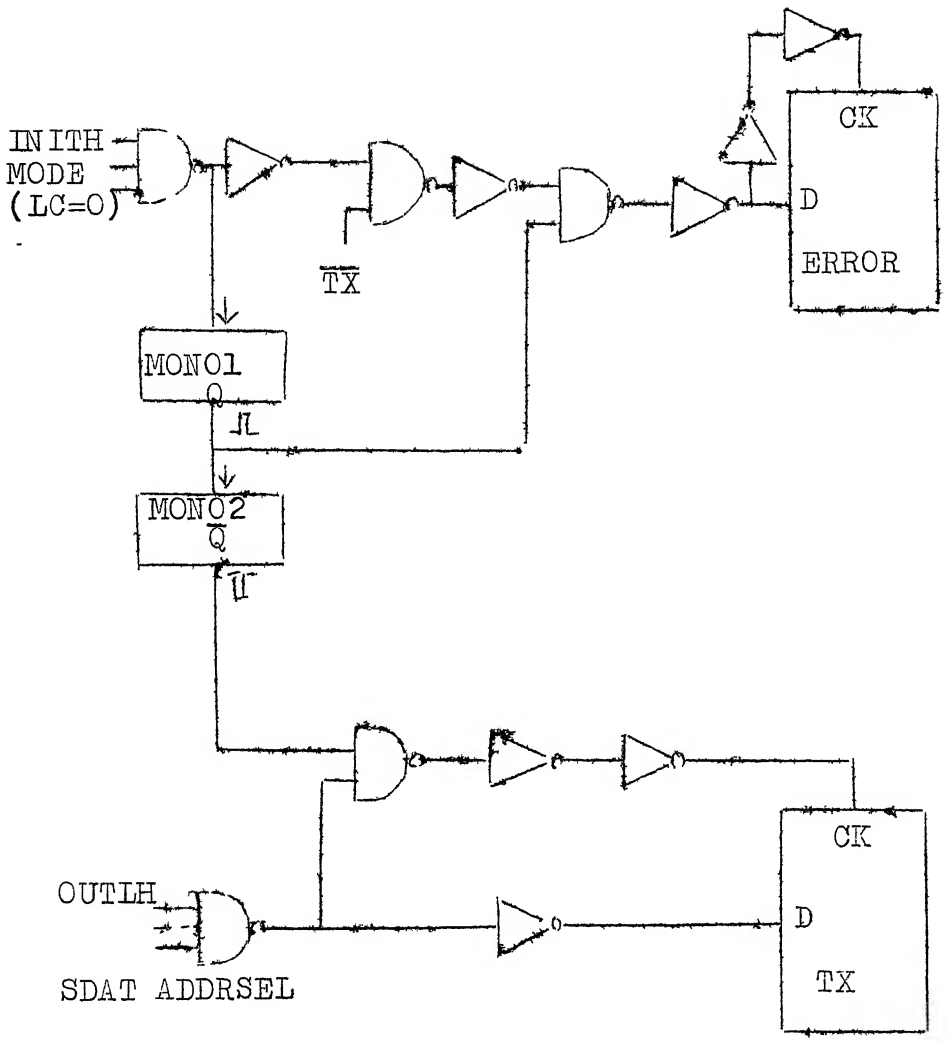


Fig. 4.10F Error and TX Control

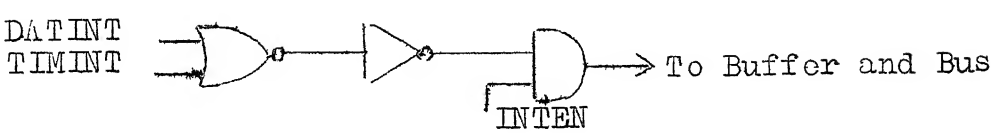


Fig. 4.11 Interrupt Generation

CHAPTER 5

RECEIVING INTERFACE

The receiving interface has the following functions.

(i) It should generate a clock that is synchronous with the serial data from the line (bit synchronisation).

(ii) It should do byte synchronisation by looking for the SYN character among the incoming bits.

(iii) It should be able to decode SOM or ACK in the idle state and generate an interrupt to the Pc.

(iv) If it is in the message reception mode, it should generate an interrupt for each byte received.

(v) It should check for parity in the received data and if in error, set a flag to inform the Pc of a parity error.

(vi) There should be provision for the receiving interface to know if for any reason, the Pc has been unable to grant its previous request and hence data overrun has occurred in the register. In such a case a flag should be set.

5.1 Hardware Description of the Receiving Interface :

As in the sending interface, the Pc can address two 8-bit registers in the receiving interface - a CSR (RCSR) and a data buffer (RDAT).

The format of the RCSR is shown in (Fig. 5.1).

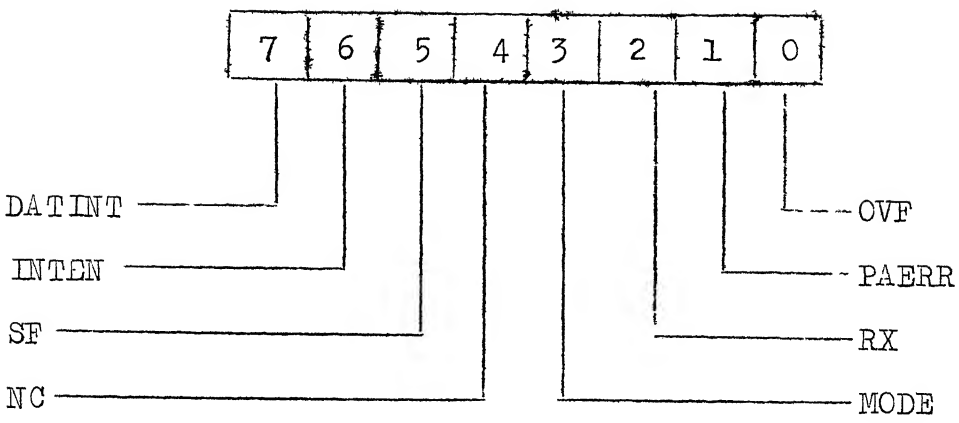


Fig. 5.1 Receiving Interface CSR

The line protocols for the receiving interface may be formulated as follows :

/$\overline{SF}$·PP/ If [RXRG = SYN] then [SF⟵1, LC⟵9] R1

/PP/ Shift right RXRG, RXRG (8)⟵ [bit from the line] R2

/[LC ≠ 0] · PP/ Decrement LC; R3

/[LC = 0] · PP/ LC⟵9 R4

/SF·$\overline{MODE}$·[LC = 0]·$\overline{PAR}$·PP/If [RXRG = SOM + ACK], then
[MODE⟵1, RDAT⟵RXRG, RX⟵1, DATINT⟵1] R5
else [if RXRG ≠ SYN then [SF⟵0]]

/SF·MODE·[LC = 0]PP/ RDAT⟵RXRG, DATINT⟵1 R6
if [PAR = 1] then PAERR⟵1
if [RX = 1], OVF⟵1; RX⟵1

/Data read by Pc from RDAT/RX⟵0 R7

/Data register read/DATINT⟵0 R8

RXRG is the receiving shift register (Fig. 5.2). Serial data from the line is inverted (inverted data is sent over the line from the sending interface), and is clocked with the bit synchronised clock PP into the RXRG. SF is the synchronisation flag and is set to 1 when byte synchronisation is obtained [refer statement R1 in receiving protocols].LC is a line counter. When SF is set, LC is initialised to 9, and then the

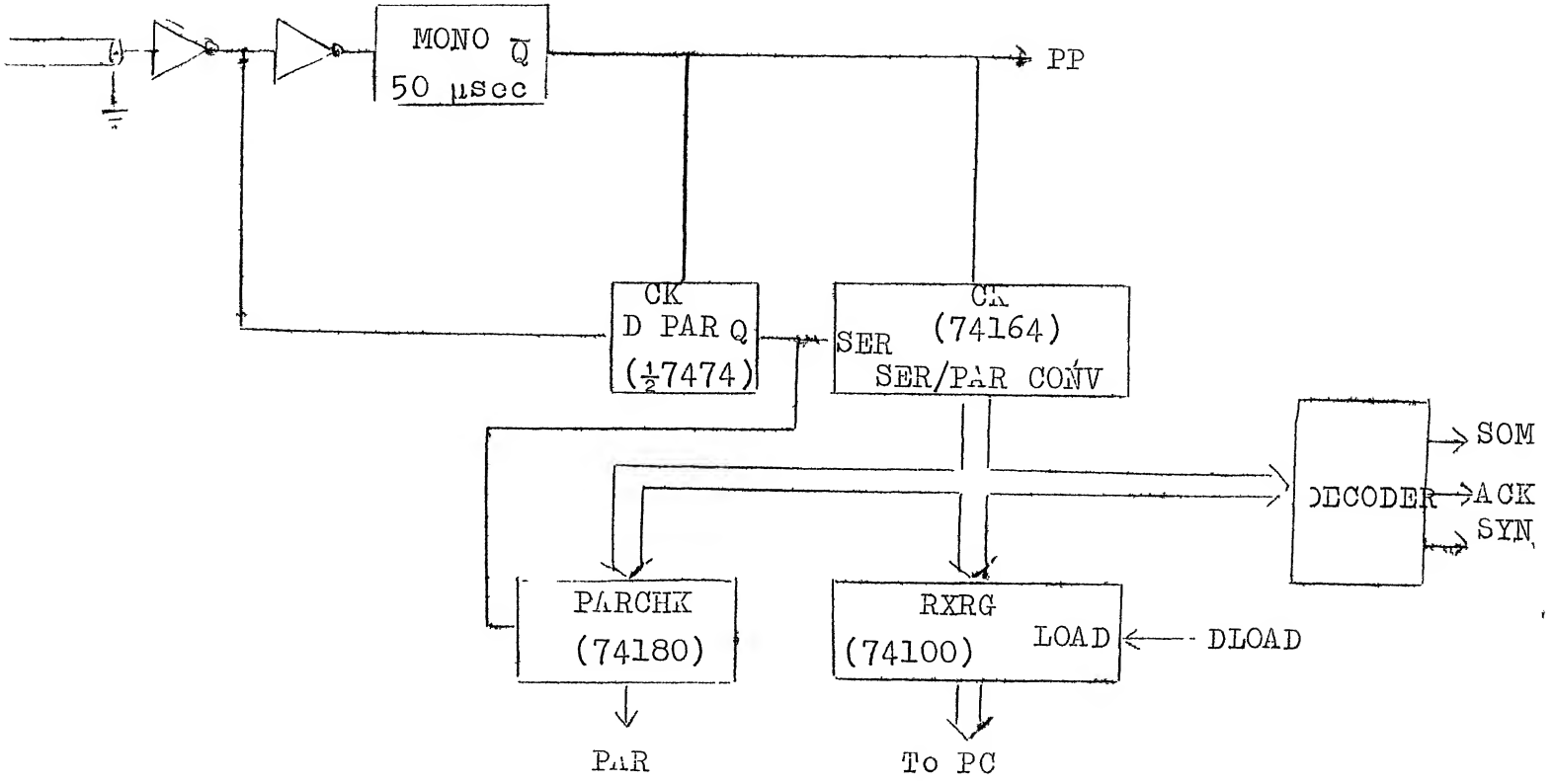


Fig. 5.2 Data Reception Block Diagram

(LC = 0) pulse would denote the arrival of a byte in RXRG. MODE bit is set to 1 when SOM or ACK is received and denotes that the receiving interface is receiving a message. RX is a data handover bit. It is set to 1 when data is loaded into the data register and is reset when data is taken in by the Pc. If at any time, a data byte is received and RX is 1, then it shows that data overflow is imminent and OVF flag gets set (R6). If a parity error is detected, PAERR flag gets set.

The hardware of the receiving interface is also designed on similar lines as that of the sending interface. Functionally, the hardware may be divided into two blocks : (i) Data - reception part (ii) CSR controls. The subblocks of the Data reception part are (A) Clock synchroniser circuit (B) Serials to parallel converter circuit (C) Byte synchroniser circuit. All these are explained in the following sections.

5.2 Clock Synchronisation Circuit :

The function of this block is to generate a clock that is synchronous with the incoming data. As said before serial-data coming over the line is coded as in (Fig. 5.3).

The type of coding mentioned above gives a negative transition at the start of each bit. So the receiving interface clock has to be synchronised to this negative edge. Two techniques are available.

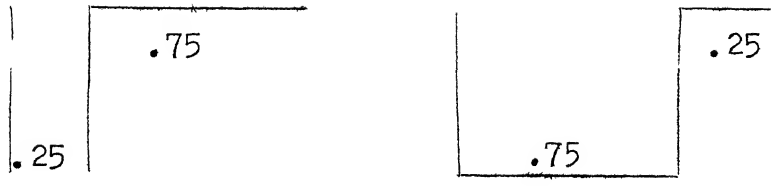


Fig. 5.3 Received Data Bits

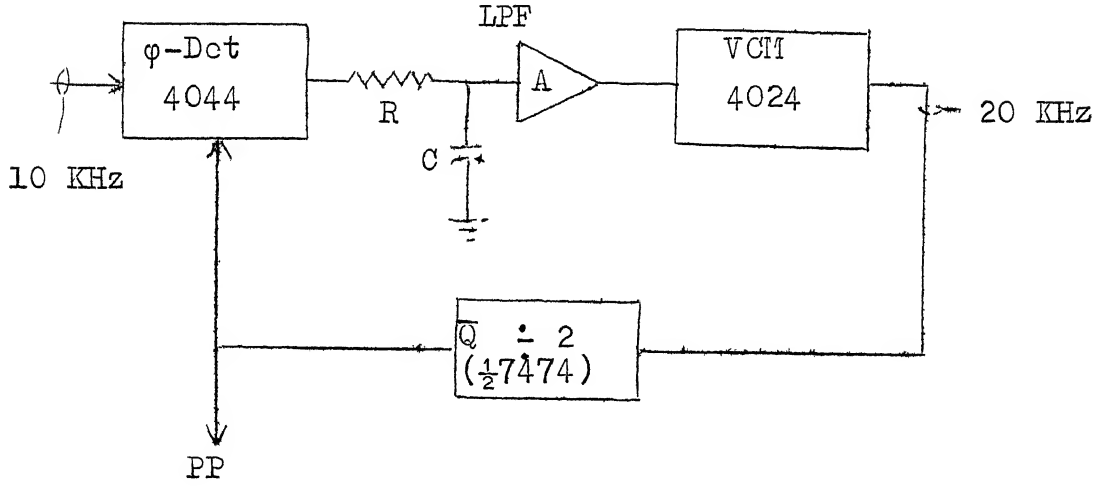


Fig. 5.4 Phase Locked Loop

(i) A Phase Locked Loop (PLL) [PHAND] could be used to generate a clock which is locked to the negative edge of the received data. One important characteristic of this PLL has to be that, it should not be sensitive to duty cycle variations (As could be seen, the duty cycles of the 1 and 0 bits are different). Such a PLL could be made using a 4044 phase detector and 4024 voltage controlled multivibrator. The circuit is shown in (Fig. 5.4).

Such a system worked well. But it was found that the VCM was very sensitive to power supply variations. Typically, the circuit locked only at 4.9 V $\pm$ .1 V. This was too critical.

(ii) Using a Monostable : This is the simplest and the crudest way of generating a synchronous clock. A monostable is tuned to a pulse-width of 50 μsec. and is triggered by the negative edge of the input bit stream. One big question is the long term stability and the short term variations in such a circuit. Regarding short term variations, it could be seen that, if data is strobed anywhere between (.25 PP) and (.75 PP) where PP is 1 clock pulse period, the stored data will be correct. This allows a 50 percent variation in pulse-width without loss of data. Short term tolerance of the monostable is much better than this. Long term stability is determined by the components. So the RC of the monostable has to be of good long term stability.

The second circuit was finally implemented since it was found to be much more reliable.

5.3 Serial/Parallel Converter :

An 8 bit shift register (74164) with a D flip-flop connected in cascade receives the input data. Finally the 8 data bits will be in the shift register and the parity bit in the D flip-flop.

5.4 Byte Synchroniser Circuit :

There is a line counter 'LC' which counts the number of bits coming into the shift register. This is a programmable UP/DOWN counter (74190), and counts down from 9 and at 0 gets loaded. The initialisation of the counter has to be done by decoding the contents of the shift-register.

As mentioned in Section 5.1, byte synchronisation in the receiving interface is indicated by the setting of the SF flag. So the condition for initialising LC is the logical function ($\overline{SF}$·SYNC·PP) where SYNC is the SYN detection pulse, $\overline{SF}$ shows that SF is not set and PP is the receiving interface clock (Fig. 5.5).

5.5 RCSR Controls :

As in the sending interface, the CSR is implemented using four 7474s. Preset and clear are used by the computer to set or clear the bits (Section 4.8). Synchronous controls D

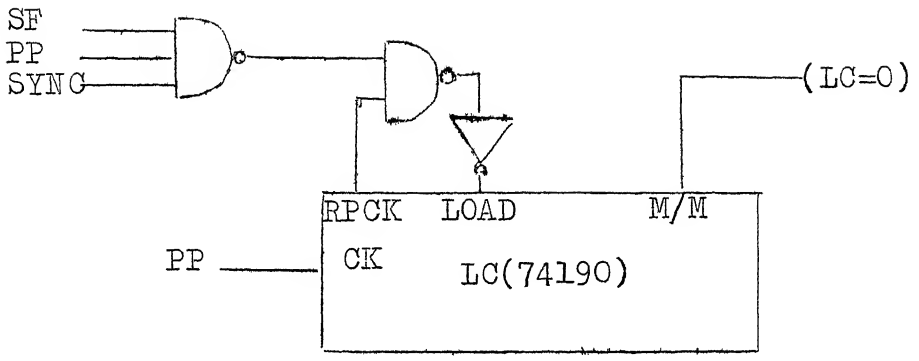


Fig.5.5 Byte Synchroniser

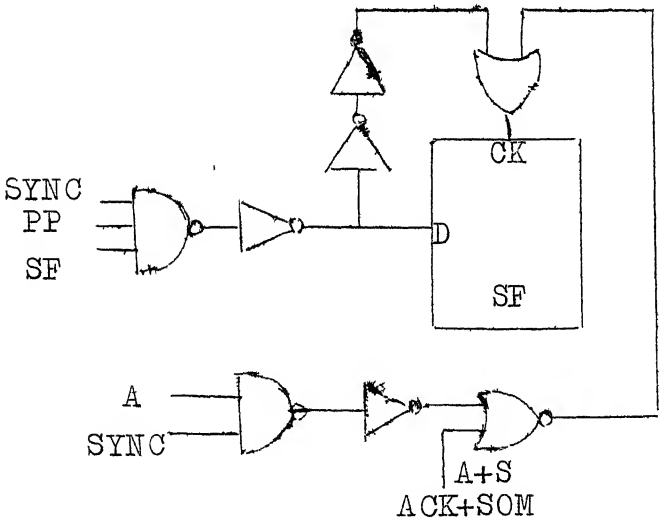


Fig. 5.6A SF Control

and CK are used by the interface. The individual controls are shown in Figs. 5.6A to 5.6D and described below :

5.5.1 SF Controls :

When SF·SYNC·PP = 1, SF is set. When MODE = 0 (idle mode) and neither SYN nor ACK nor SOM comes over the line SF is reset. Signal A is SF·(LC = 0) · $\overline{MODE}$ $\overline{PAR}$

5.5.2 MODE and DATINT Controls :

MODE is set when SF·(LC = 0)·$\overline{MODE}$·$\overline{PAR}$·(ACK + SOM) = 1. It is reset by software.

DATINT gets set when DATEN is set and after data register RDAT gets loaded (DDLD is a delayed version of DLOAD which loads RDAT). It also gets set by the same signal that sets MODE. DATINT is reset when RDAT is read by software.

5.5.3 PAERR Control :

PAERR gets set when a parity error is detected by the parity checker (Fig. 5.2). PAR output is the output of the parity checker which is 1 when there is a parity error in the received data.

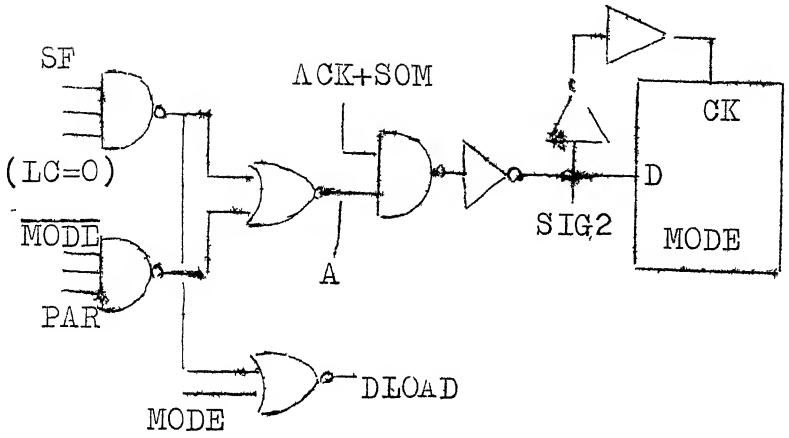
5.5.4 RX and OVF Control :

RX is set to 1 when RDAT is loaded by DLOAD. If the interrupt is granted and the RDAT is read, by the Pc, RX gets reset.

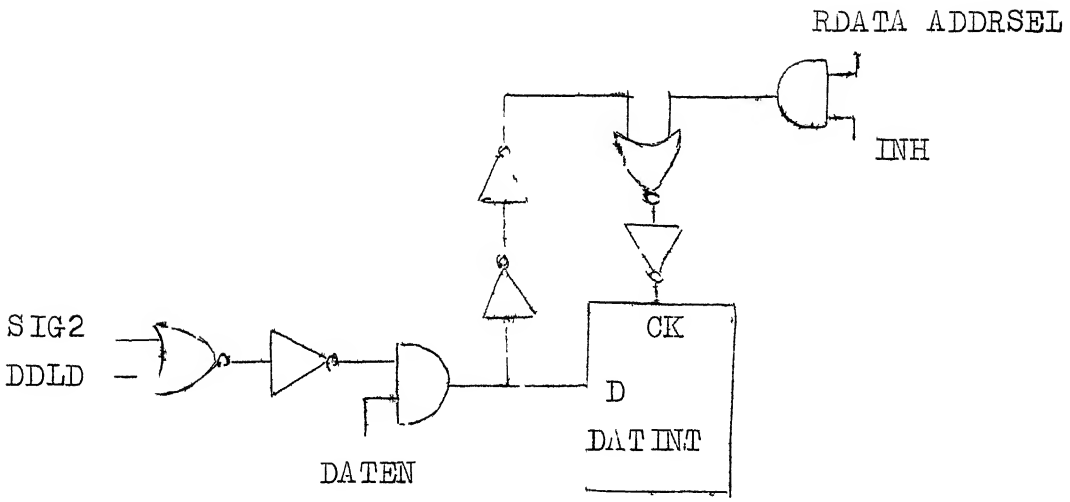
If RX still is set if MONOI is active, it means the previous interrupt has not been granted and OVF gets set.

5.6 Interrupt Generation :

Interrupt is generated when DATINT and DATEN are set.



(a) MODE Control



(b) DATINT Control

Fig. 5.6B MODE and DATINT Control

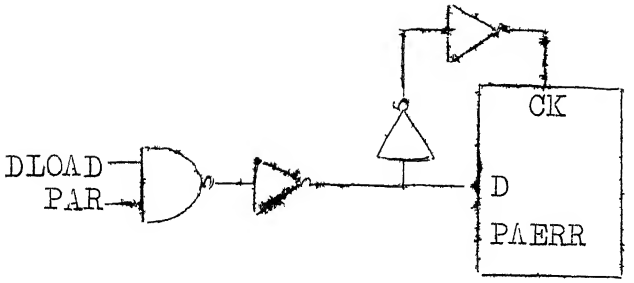


Fig. 5.6C PAERR Control

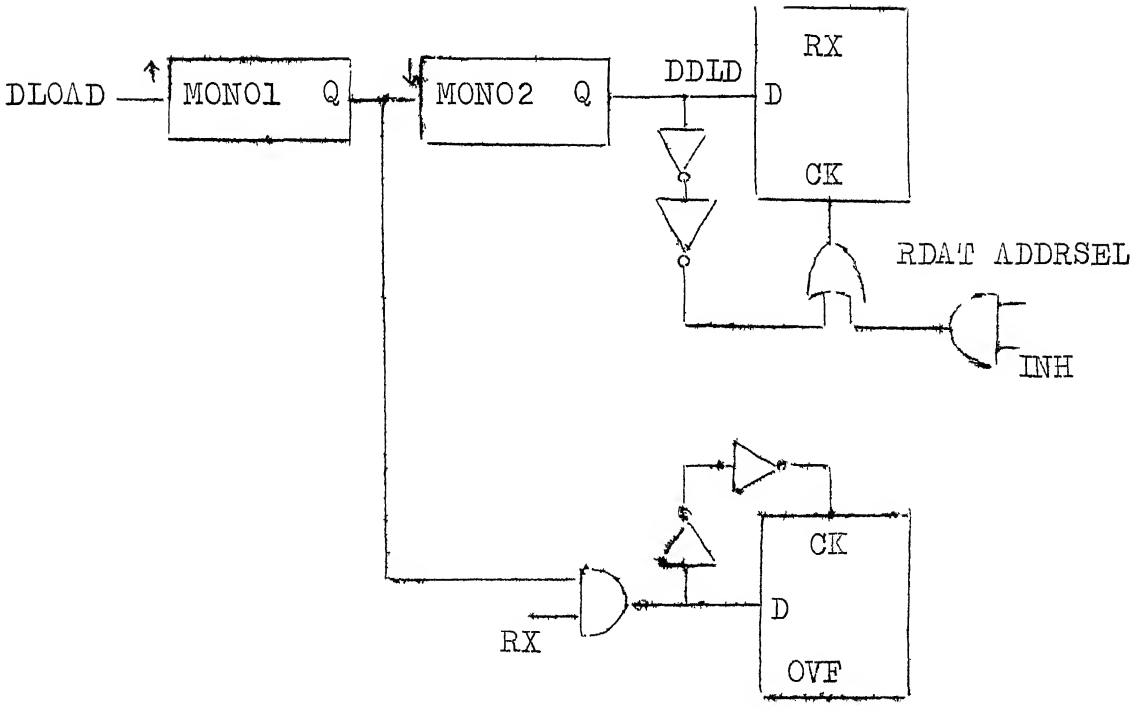


Fig. 5.6D RX and OVF Control

CHAPTER 6

CONCLUSION

The aim of the project was to link up TDC-316, IBM-1800 and DEC-1090 in a star network with MICRO-78 as the switching centre. Work on the MICRO-78 interface to the DEC-1090 could be started only very recently since DECNET protocols arrived late. Also, the MICRO-78 which was supposed to be at the nodal point has not yet arrived. So initial testing of all the three interfaces for TDC-316, IBM-1800 and MICRO-78 was done by connecting the sending interfaces of the machines to their own receiving interfaces. This loop-back testing was successful.

Since TDC-316 and IBM-1800 interfaces were already operational in the Computer Centre, they were directly connected together and messages transferred between them. All bit combinations from 0 to 377/8 were generated, transmitted, and received, without any error. Any further work can be done only after MICRO-78 arrives and DEC-1090 hardware is ready. Thus the network will be hopefully completely functional by July, 1980.

The following points came out when recapitulating on the various aspects of the system design.

1. From the protocols, it could be seen that ACK and SOM are treated identically by the Pc. Both interrupt the Pc and further action is taken by the Pc after identifying ACK or SOM through software. Immediately the question arises whether it was necessary to have an ACK character at all. Why not recognise only SOM and decide to send acknowledgement as any other message? In fact, in the TDC-316 hardware interface, it is possible to treat ACK also as a message. Thus, recognition of ACK by hardware is a redundant feature.

2. There is no flexibility as far as data rate is concerned. Some hardware changes will be necessary if data rate is to be varied. In the TDC interface, it would be possible to send data at any frequency from 5 KHz to 50 KHz at steps of 5 KHz by setting switches. In the receiving interface, monostable pulse-width will have to be changed accordingly, by varying the RC time constant.

Future work on this project can proceed in two directions -

A. The hardware development could be done by implementing a DMA interface for MICRO-78 and TDC-316. With the DMA interface implementation, it should be possible to go to much higher data rates.

B. It should be possible to modify the present hardware to form a data loop. This would be complicated by the fact that there is provision only for two synchronous lines on the DEC-10 side. It would need a sort of direct linking between the sending interface of any Pc to its own receiving interface and also decoding circuits which could identify the source and the destination of the packet received.

A lot of work can be done on the software side. The software work could start with the design of a Network Operating System, along with a language for designing the O.S. These could be linked to the KDM of TDC-316. Also performance studies could be conducted on the network eg: effect of packet length on error rates, processor utilisation studies etc.

This network can be further extended by connecting a separate MICRO-78 at each nodal computer that can serve as a communication processor. The nodal computer communicates with its communication processor which in turn can communicate with the central switch. By this method, we have two more nodal points at each communication processor which can be connected to other computers or MICRO-78s for the enlargement of the network.

REFERENCES

[ABRA and KUO] Abramson, N. and Kuo, F.F. (eds): Computer Communication Networks (1973).

[BRAN-72] Brandt, G.J. and Chretien, G.J.: Methods to Control and Operate a Message Switching Network in Proceedings of the Symposium on Computer Communications Networks and Teletraffic (1972), 263-276.

[CHU] Chu, Yaohan :'Computer Organisation and Microprogramming', Prentice-Hall, 1972.

[CROW-75] Crowther, W.R., et al: Issues in Packet Switching Network Design: Proc. AFIPS NCC 44(1975), 161-175.

[CVK-79] C.V. Krishna,: IIT/K Computer Network - IBM-1800 Hardware Interface: M.Tech. Thesis.

[DAVI-68] Davies, D.W: 'The Principles of a Data Communication Network for Computers and Peripherals' Advances in Computer Communications (1974) Edited by Chu, W.W.

[DAVI and BARB] Communication Networks for Computers (1973), Davies, D.W., and Barber, D.L.A.

[DDAS-79] Debasis Das: IIT/K Computer Network - MICRO-78 Hardware Interface: M.Tech. Thesis.

[FRAN-72] Frank, H., and Chou, W. : Topological Optimisation of Computer Networks : Proc. IEEE 60, 11(Nov. 1972), 1385-1397.

[FRAS-76] Fraser, A.G. : The Present Status and Future Trends in Computer Communication Technology : Computer (Sept, 1976), 10-19.

[GRAY-72] Gray, J.P. : Line Control Procedures : Proc. IEEE 60, 11 (Nov. 1972), 1301-1312.

[KAHN-72] Kahn, R.E. : Resource Sharing Computer Communications Networks : Proc. IEEE 60, 11(Nov.1972), 1397-1407.

[KLEI] Kleinrock, L. : Queuing Systems, Volume II, Computer Applications (1976).

[ORNS-72] Ornstein, S.M., et al : The Terminal IMP for the ARPA Computer Network: Proc. AFIPS SJCC 40(1972), 243-254.

[POUZ-74] Pouzin, L. : Cigale, the Packet Switching Machine for the CYCLADES Computer Network: Proc. IFIP, Stockholm (Aug. 1974), 155-159.

[STUT-72] Stutzman, B.W. : Data Communication Control Procedures : Computing Surveys 4, 4(Dec.1972), 197-220.

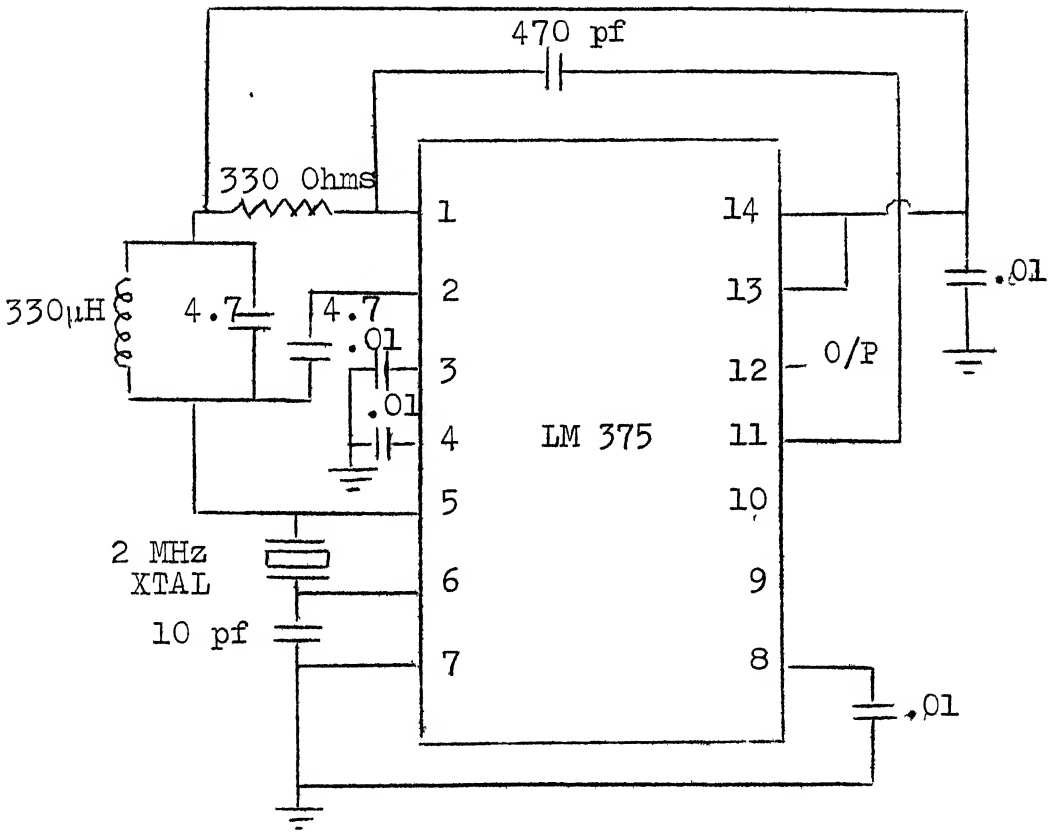
[TDCSYSI] TDC-316 System Manual, Vol. I.

[TTLH] Texas Instruments Inc: 'Designing with TTL Integrated Circuits', ISE, McGraw-Hill, 1971.

[WALD-75] Walden, D.C.: Experience in Building, Operating, and Using the ARPA Network: Second USA - Japan Comp. Conf., Tokyo (Aug. 1975).

[PHAND] Phase Locked Loop Systems, DATA BOOK, MOTOROLA Semiconductor Products Inc., Aug. 1973.

Appendix A-1



LM 375 Clock Generator